**MoSCoW Prioritisation**

SHOULD HAVE

* A statistical analysis tool (SonarQube) ran through the coding, with relevant refactoring according to the analysis.
* Mockito Testing for the Service Database class.
* A H2 console for testing,
* At least 5-7 risks within the risk assessment.
* Relationship of entity modelled using an ERD.
* Before-and-after ERD’s and UML.
* Unit test coverage of CRUD functionality of the src/main/java folder, aiming for 80%.
* An adherence to best practice (OOP principles, SOLID, refactoring)
* A working “.ignore” for ignoring build-generated files and folders
* A postman application to check the http requests.

COULD HAVE

WON’T HAVE

* Extra functionality outside of the specification.
* Unnecessary imports / dependencies / database tables.
* More than 1 Entity due to the timeframe.
* Complex CSS styling.
* Any incomplete documentation.

MUST HAVE

* A build of the application present in the root folder.
* A working application that is interactive with a webpage (front-end)
* Back-end application developed using Java (Spring Boot Framework)
* Integration and Unit tests for validation of the application. (Junit & Mockito)
* A managed MySQL database connected locally.
* A fully functioning CRUD application following the project model.
* A front-end developed using JavaScript, HTML and CSS.
* A completed project management board; user stories, acceptance criteria + story points estimation.
* Code fully integrated into a Version Control System (Git), employing a Feature-Branch Model.
* A risk assessment outlining the risks during the project timeframe (pdf).
* A relational database used to store data for the project with at least 1 entity.
* At least one ERD and one UML diagram (png)
* A completed README.md, explaining how to use and test the application.
* A copy of the presentation (pdf)
* A functional application created in the OOP language
* A functional application front-end which connects to the back-end API.
* An application build, including possible dependencies, produced with integrated build tool (Maven).
* Must meet 80% test coverage of the backend.
* Selenium for automated front-end testing.
* More than 1 entity; like a Pokémon Trainer entity tied to the Pokémon entity.
* The project connecting via JDBC to a GCP-based MySQL instance.
* Test coverage of CRUD Functionality above 80%.
* An evolution of ERD/UML Diagrams over time.
* A log-in page for user (trainers).